shape3d.py

```python
from utils import *
from shape2d import Vertex, Shape2D

class Shape3D:
    def __init__(self, _position : Vertex = Vertex(0, 0, 0), _vertices : list[Vertex] = list(), _surfaces : list[Shape2D] = list(), _color : tuple[float] = None, _direction : list[float] = None) -> None:
        self.position : Vertex = _position
        self.vertices : list[Vertex] = _vertices
        self.surfaces : list[Shape2D] = _surfaces
        self.color: tuple[float] = _color
        self.direction : list[int] = [uniform(*UNIFORM_DIRECTION_AREA), uniform(*UNIFORM_DIRECTION_AREA), uniform(*UNIFORM_DIRECTION_AREA)] if not _direction else _direction

    def darken_color(self, _color, _delta = 0.2):
        return (_color[0] - _delta, _color[1] - _delta, _color[2] - _delta)

    def draw(self, _object_position : Vertex, _draw_edges : bool = False, _draw_vertices : bool = False) -> None:
        for surface in self.surfaces:
            if _draw_edges:
                for i in range(len(surface.vertices) - 1):
                    glLineWidth(2)
                    glBegin(GL_LINES)
                    glColor3f(*self.darken_color(surface.color))
                    glVertex3f(
                        _object_position.x + surface.vertices[i].x,
                        _object_position.y + surface.vertices[i].y,
                        _object_position.z + surface.vertices[i].z
                    )
                    glVertex3f(
                        _object_position.x + surface.vertices[i + 1].x,
                        _object_position.y + surface.vertices[i + 1].y,
                        _object_position.z + surface.vertices[i + 1].z
                    )
                    glEnd()
                    glLineWidth(1)
            surface.draw(_object_position, _draw_vertices)

    def translate(self, _vector : list[float]) -> None:
        for vertex in self.vertices:
            vertex.translate(_vector)
        self.position.translate(_vector)

    def rotate(self, _object_position : Vertex, _angle : float, _axis : str) -> None:
        for vertex in self.vertices:
            vertex.rotate(_object_position, _angle, _axis)
        self.position.rotate(_object_position, _angle, _axis)


    def rotate_any(self, _object_position : Vertex, _yaw : float, _pitch : float, _roll : float) -> None:
        for vertex in self.vertices:
            vertex.rotate_any(_object_position, _yaw, _pitch, _roll)
        self.position.rotate_any(_object_position, _yaw, _pitch, _roll)

```